

Our Ref. No.042390.P13464
Express Mail No.: EL 802886885 US

UNITED STATES PATENT APPLICATION

FOR

POWER MANAGEMENT USING PROCESSOR THROTTLING EMULATION

Inventor:

Barnes Cooper

Prepared by:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
12400 Wilshire Blvd., 7th Floor
Los Angeles, CA 90025-1026
(714) 557-3800

POWER MANAGEMENT USING PROCESSOR THROTTLING EMULATION

BACKGROUND

1. Field of the Invention

[001] This invention relates to microprocessors. In particular, the invention relates to power management.

2. Description of Related Art

[002] Mobile platforms have historically been faced with power and thermal management challenges due to the small form factor, limited power source availability, and relatively high power processors. Techniques have been developed to reduce processor power dissipation.

[003] One technique is to use throttling a stop clock (STPCLK#) signal to the processor. When STPCLK# signal is asserted to the processor, the processor enters a low power state such as Mobile Quick Start or stop grant. When STPCLK# is de-asserted, the processor resumes execution. Throttling using STPCLK# involves modulating STPCLK# signal at a specific frequency and duty cycle, whereby effectively the processor operates at a reduced frequency. Throttling using STPCLK# has a number of drawbacks. First, it is complicated and requires complex external interface circuits and protocol. Second, it is not compatible with software standards in power management. Third, it is not efficient. Fourth, it is not flexible in generating an arbitrary duty cycle for the processor state.

[004] Therefore, there is a need to have a technique for power management that can overcome the above drawbacks.

BRIEF DESCRIPTION OF THE DRAWINGS

[005] The features and advantages of the present invention will become apparent from the following detailed description of the present invention in which:

[006] Figure 1 is a diagram illustrating a system in which one embodiment of the invention can be practiced.

[007] Figure 2 is a diagram illustrating a throttling emulator according to one embodiment of the invention.

[008] Figure 3 is a flowchart illustrating a process to initialize timer-based power state change according to one embodiment of the invention.

[009] Figure 4 is a flowchart illustrating a process to emulate throttling using timer according to one embodiment of the invention.

[0010] Figure 5 is a diagram illustrating a duty cycle of the processor state according to one embodiment of the invention.

DESCRIPTION OF THE INVENTION

[0011] The present invention is a technique to emulate throttling a processor for power management. The technique includes (1) determining a processor state, which may be an operational state or a low power state, of a processor upon expiration of a system management interrupt (SMI) timer, (2) loading the SMI timer with a timer value based on the processor state, and (3) transitioning the processor to one of the operational state and the low power state according to the processor state. The timer value may be a first value and a second value which define the duty cycle of the processor state.

[0012] In the following description, for purposes of explanation, numerous details are set forth in order to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that these specific details are not required in order to practice the present invention. In other instances, well-known structures are shown in block diagram form in order not to obscure the present invention.

[0013] The present invention may be implemented by hardware, software, firmware, microcode, or any combination thereof. When implemented in software, firmware, or microcode, the elements of the present invention are the program code or code segments to perform the necessary tasks. A code segment may represent a procedure, a function, a subprogram, a program, a routine, a subroutine, a module, a software package, a class, or any combination of instructions, data structures, or program statements. A code segment may be coupled to another code segment or a hardware circuit by passing and/or receiving information, data, arguments, parameters, or memory contents. Information, arguments, parameters, data, etc. may be passed, forwarded, or transmitted via any suitable means including memory sharing, message passing, token passing, network transmission, etc. The program or code segments may be stored in a processor readable medium or transmitted by a computer data signal embodied in a carrier wave, or a signal modulated by a carrier, over a transmission medium. The "processor readable medium" may include any medium that can store or transfer information. Examples of the processor readable medium include an electronic circuit, a semiconductor memory device, a ROM, a flash memory, an erasable programmable ROM (EPROM), a floppy diskette, a compact disk (CD-ROM), an optical disk, a hard disk, a fiber optic medium, a radio frequency (RF) link, etc. The computer

data signal may include any signal that can propagate over a transmission medium such as electronic network channels, optical fibers, air, electromagnetic, RF links, etc. The code segments may be downloaded via computer networks such as the Internet, Intranet, etc.

[0014] It is noted that the invention may be described as a process which is usually depicted as a flowchart, a flow diagram, a structure diagram, or a block diagram. Although a flowchart may describe the operations as a sequential process, many of the operations can be performed in parallel or concurrently. In addition, the order of the operations may be re-arranged. A process is terminated when its operations are completed. A process may correspond to a method, a function, a procedure, a subroutine, a subprogram, etc. When a process corresponds to a function, its termination corresponds to a return of the function to the calling function or the main function.

[0015] Figure 1 is a diagram illustrating a system 100 in which one embodiment of the invention can be practiced. The system 100 includes a host processor 110, a host bus 120, a memory control hub (MCH) 130, a basic input and output system (BIOS) 135, a system memory 140, an input/output control hub (ICH) 150, a peripheral bus 155, a mass storage device 170, and input/output devices 180₁ to 180_K. Note that the system 100 may include more or less elements than these elements.

[0016] The host processor 110 represents a central processing unit of any type of architecture, such as embedded processors, mobile processors, micro-controllers, digital signal processors, superscalar computers, vector processors, single instruction multiple data (SIMD) computers, complex instruction set computers (CISC), reduced instruction set computers (RISC), very long instruction word (VLIW), or hybrid architecture. In particular, the processor 110 includes a performance control (PERF_CTL) register 102 and a performance status register (PERF_STS) 105. The PERF_CTL register 102 allows an OS to control the processor performance by changing the bus ratio and operating voltage. The bus ratio is related to the operating frequency and the voltage is related to the power consumption. The PERF_STS register 107 stores the current bus ratio and the current voltage identifier which is used to control a voltage regulator to generate appropriate operating voltage.

[0017] The host bus 120 provides interface signals to allow the processor 110 to communicate with other processors or devices, e.g., the MCH 130. The host bus 120 may support a uni-processor or multiprocessor configuration. The host bus 120 may be parallel, sequential, pipelined, asynchronous, synchronous, or any combination thereof.

[0018] The MCH 130 provides control and configuration of memory and input/output devices such as the BIOS 135, system memory 140, and the ICH 150. The MCH 130 may be integrated into a chipset that integrates multiple functionalities such as the isolated execution mode, host-to-peripheral bus interface, memory control. The MCH 130 interfaces to the peripheral bus 155. For clarity, not all the peripheral buses are shown. It is contemplated that the system 100 may also include peripheral buses such as Peripheral Component Interconnect (PCI), accelerated graphics port (AGP), Industry Standard Architecture (ISA) bus, and Universal Serial Bus (USB), etc.

[0019] The BIOS 135 stores boot-up or initialization code and data including look-up tables. The BIOS 135 is typically implemented with non-volatile memories such as flash memory, read only memory (ROM), erasable ROM, etc. The BIOS 135 or part of it may also be located internally to the MCH 130 or ICH 150.

[0020] The system memory 140 stores system code and data. The system memory 140 is typically implemented with dynamic random access memory (DRAM) or static random access memory (SRAM). The system memory may include program code or code segments implementing one embodiment of the invention. The system memory includes a throttling emulator 145 which may include separate elements. Any one of the elements of the thermal management module 145 may be implemented by hardware, software, firmware, microcode, or any combination thereof. The system memory 140 may also include other programs or data which are not shown, such as an operating system. The throttling emulator 145, when executed, causes the processor 110 to perform a number of tasks or operations as described later.

[0021] The ICH 150 has a number of functionalities that are designed to support I/O functions. The ICH 150 may also be integrated into a chipset together or separate from the MCH 130 to perform I/O functions. The ICH 150 may include a number of interface and I/O functions such as PCI bus interface to interface to the peripheral bus 155, processor

interface, interrupt controller, direct memory access (DMA) controller, power management logic, timer, system management bus (SMBus), universal serial bus (USB) interface, mass storage interface, low pin count (LPC) interface, etc. In one embodiment, the ICH 150 has support for ACPI operations including system management interrupt (SMI) and system control interrupt (SCI).

[0022] The mass storage device 170 stores archive information such as code, programs, files, data, applications, and operating systems. The mass storage device 170 may include compact disk (CD) ROM 172, floppy diskettes 174, and hard drive 176, and any other magnetic or optic storage devices. The mass storage device 170 provides a mechanism to read machine-readable media. The machine-readable media may contain computer readable program code to perform tasks as described in the following. These tasks may include determining a processor state of a processor upon expiration of a system management interrupt (SMI) timer, loading the SMI timer with a timer value based on the processor state, and transitioning the processor to an operational state or a low power state according to the processor state.

[0023] The I/O devices 180₁ to 180_k may include any I/O devices to perform I/O functions. Examples of I/O devices 180₁ to 180_k include controller for input devices (e.g., keyboard, mouse, trackball, pointing device), media card (e.g., audio, video, graphics), and any other peripheral controllers.

[0024] The embedded controller (EC) 190 is any controller such as micro-controller, digital signal processor, or any programmable device that can execute its own programs. The embedded controller 190 contains EC program code 195. The EC program code contains instructions that cause the EC 190 to perform specified operations.

[0025] Figure 2 is a diagram illustrating the throttling emulator 145 according to one embodiment of the invention. The throttling emulator 145 may include a number of elements that belong to separate modules. For clarity and illustrative purposes, the throttling emulator 145 is shown to be located in the system memory. Components of the throttling emulator 145 may be located in other places. In addition, any one of the components of the throttling emulator 145 may be implemented by software, firmware, hardware, or any combination thereof. The throttling emulator 145 includes a system

management interrupt (SMI) handler 210, a throttling state 220, an Advanced Configuration and Power Interface (ACPI) Operating System (OS) 230, an SMI timer 240, a timer enable selector 245, an SMI timer handler 250, a low power state 260, and an operational state 270.

[0026] The SMI handler 210 interfaces with the ACPI OS 230 and the SMI timer 240. It is a function or routine that is invoked in response to an SMI or an input/output (I/O) trap generated by a chipset (e.g., the ICH 150 in Figure 1). The ACPI OS 230 receives a dummy address to enable the SMI handler to trap on this dummy address. The dummy address is a false address with respect to the correct address of the throttling register or state 220. The purpose of providing a dummy address is to avoid the ACPI OS 230 from directly manipulating hardware registers that may not function properly or are not implemented or where the platform designers may wish to employ alternative behavior or functionality, etc. The SMI handler 210 also boots the ACPI OS 230 by loading the ACPI OS 230 into the system memory 140 (Figure 1). The SMI handler 210 interfaces to the SMI timer 240 to enable/disable the SMI timer 240 to enable/disable the throttling emulation. If the throttling emulation is enabled, the SMI handler 210 initializes the SMI timer 240 by loading the initial value to the SMI timer 240.

[0027] The throttling state 220 THROT includes throttling control information such as the throttling registers used to emulate the processor throttling through the SMI handler SMI trap on the dummy address. The throttling state 220 is pointed to by the dummy address. In essence, the throttling state 220 implements a data structure similar to the description table (e.g., the FADT) and the associated registers which the ACPI OS 230 would have accessed in a normal power management scheme. This data structure may include throttling parameters such as the duty cycle of the processor state, and the processor state to be transitioned to. By re-directing the access to the throttling state 220 instead of the actual FADT, the technique provides direct control on the throttling function as if it were performed using the standard technique of using the STPCLK# signal.

[0028] The ACPI OS 230 is the OS that is compatible to the power management scheme as specified in the ACPI standard, published by Compaq Computer Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd, and Toshiba Corporation, Revision 2.0, in July 27 2000. The ACPI OS 230, when booted by the SMI handler 210,

obtains the dummy address and accesses the throttling state 220 to set the control parameters for power management. These control parameters may include the duty cycle of throttling, the power state, and the latency.

[0029] The SMI timer 240 generates periodic SMI timer event to the SMI timer handler 250 when it expires. Alternatively, the SMI timer 240 may generate SMI timing event to the SMI handler 210 which in turn invokes the SMI timer handler 250. The SMI timer 240 may be any suitable timing device such as the legacy timer in the chipset ICH 150 (Figure 1). The SMI timer is enabled/disabled by the SMI handler 210 and the SMI timer handler 250 via the enable selector. The SMI timer event may be programmed to wake the processor from a low power state. Initially, the SMI timer 240 is enabled and initialized by the SMI handler 210. Once initialized, the SMI timer 240 runs independently and generates an SMI timer event upon expiration, i.e., when it reaches the terminal count. It can then be reloaded by new value to start another period until the next expiration. By alternately loading two timer values into the SMI timer 240, a variable duty cycle for the processor state can be achieved.

[0030] The SMI timer handler 250 is a function or routine to control the SMI timer 240 to transition the processor into one of the low power state 260 and the normal operational state 270 for periods corresponding to the timer values to be loaded into the SMI timer 240. The SMI timer handler 250 is invoked by the SMI handler 210 to service the SMI timer event provided by the SMI timer 240 at timer expiration.

[0031] The low power state 260 represents the power states that the processor is transitioned to. Examples of the low power state 260 include the power states C1, C2, C3, and the sleep state S1 as described in the ACPI standard version 2.0. To enter these power states, the SMI timer handler 250 may execute an instruction (e.g., the HLT instruction), or read a command register (e.g., the P_LVL2, P_LVL3) in accordance to the protocol set forth in the ACPI standard version 2.0.

[0032] The normal operational state 270 represents the operational state of the processor, such as the power state C0 as specified in the ACPI standard version 2.0. The operational state 270 is the state where the processor normally executes instructions. This state can be

entered by generating an interrupt to the processor. The SMI timer event may be programmed to bring the processor to the normal operational state.

[0033] Figure 3 is a flowchart illustrating a process 300 to initialize timer-based power state change according to one embodiment of the invention.

[0034] Upon START, the process 300 reports a throttling state at the dummy or false address (Block 310). The dummy address corresponds to the location where the throttling state is stored. Next, the process 300 enables the SMI handler to trap on the dummy address (Block 315). Then, the SMI handler boots the ACPI OS (Block 320) and passes the dummy address to the ACPI OS.

[0035] Next, the ACPI OS obtains the dummy address and uses it to access the throttle state (Block 325). Then, the ACPI OS accesses the throttling state as it were accessing the actual ACPI structures such as the FADT, the power registers, etc. (Block 330). Next, the chipset generates an I/O trap SMI to the SMI handler (Block 335). The SMI handler then executes the I/O trap for accessing the throttling state (Block 340).

[0036] Next, the SMI handler determines if the access to the throttling state is a read or a write access (Block 345). If it is a read access, the SMI handler returns the throttling state as the state variable THROT (Block 350) and goes to Block 375. If it is a write access, the SMI handler updates the throttling state with the new throttled state THROT (Block 355). Then, the SMI handler determines if the throttling is enabled or disabled (Block 360). If the throttling is disabled, the SMI handler disables the SMI timer, exits (Block 375) and is then terminated. If the throttling is enabled, the SMI handler enables the SMI timer and initializes the SMI timer by loading the initial timer value (Block 370). This initial timer value may be any value to allow the SMI timer to expire to generate the first SMI timer event. Then, the SMI handler exits (Block 375) and the process 300 is terminated.

[0037] Figure 4 is a flowchart illustrating a process 400 to emulate throttling using timer according to one embodiment of the invention.

[0038] Upon START, the process 400 determines if the SMI timer expires (Block 410). Typically, the SMI timer expires by an asynchronous hardware mechanism and there is no need to poll the status of the SMI timer. If the SMI timer has not expired, the process 400

is terminated. If the SMI timer expires, an SMI timer event is generated and the SMI timer handler is invoked and executed (Block 415).

[0039] Next, the SMI timer handler determines if throttling is enabled (Block 420). If not, the SMI timer handler disables the SMI timer (Block 425), exits (Block 460) and is then terminated. If throttling is enabled, the SMI timer handler determines if the processor is in normal operational state or in low power state (Block 430). If the processor is in operational state, the SMI timer handler reloads the SMI timer with a value TLOW which corresponds to the time interval that the processor is in the low power state (Block 435). Then, the SMI timer handler performs operations to enter the desired low state (Block 440). As discussed above, the low state may be any one of a power state C1, C2, C3, and the sleep state S1. Next, the SMI timer handler exits (Block 460) and the process 300 is terminated. If the processor is not in the operational state, i.e., it is in the low power state, the SMI timer handler reloads the SMI timer with a value THIGH which corresponds to the time interval that the processor is in the operational state (Block 445). Then, the SMI timer handler performs operations to enter the normal operational state (e.g., the C0 state) (Block 450). This can be accomplished by generating an interrupt to the processor. Next, the SMI timer handler exits (Block 460) and the process 300 is terminated.

[0040] Figure 5 is a diagram illustrating a duty cycle of the processor state according to one embodiment of the invention.

[0041] The processor state is either low power state or operational state. For illustrative purposes, the operational state is shown as a high value and the low power state is shown as a low value. The SMI timer event is generated when the SMI timer expires. Suppose initially the processor state is operational state. During this time, the timer has been loaded with the THIGH value. When the timer expires, it is reloaded with the TLOW value and the processor is transitioned to one of the low power states. The processor remains in the low power state for the time interval corresponding to TLOW. When the timer expires again, it is reloaded with the THIGH value and the processor is transitioned to the operational state. The processor remains in the operational state for the time interval corresponding to THIGH. Then, the timer expires and the process is repeated.

[0042] By selecting appropriate values of TLOW and THIGH, virtually any desired duty cycle can be achieved. This technique therefore provides more flexibility than the standard throttling technique where there is only a limited number of duty cycles (e.g., 8). The duty cycle is the ratio between the THIGH and the sum of TLOW and THIGH, expressed as a percentage. For example, when $TLOW = THIGH$, a 50% duty cycle results.

[0043] While this invention has been described with reference to illustrative embodiments, this description is not intended to be construed in a limiting sense. Various modifications of the illustrative embodiments, as well as other embodiments of the invention, which are apparent to persons skilled in the art to which the invention pertains are deemed to lie within the spirit and scope of the invention.